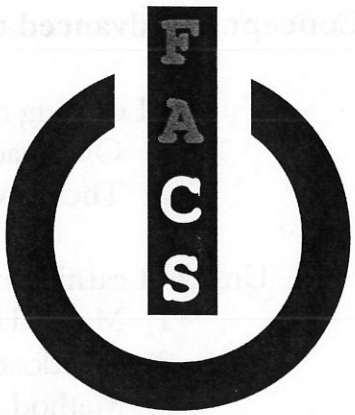
AP Computer Science

"Advanced Java Programming and Software Development"

AP Computer Science Curriculum Outline
Object Oriented Programming in Java
Fort Atkinson High School
Instructor: Dean Johnson



turnOnThePower

Concept 1: All about Classes and Objects

Unit 1: What is a java Class?

1) Observing the world around us (in-school field trip)
2) Getting the idea of a Class (no computer)
3) Generating ideas for different Classes
4) Designing a Class (group)
5) Designing a Class (individual)
6) Finding classes in the real world

Unit 2: How to write a simple java class using Dr. Java

1) Anatomy of a class
2) Class Header
3) Empty Constructor
4) Instance Fields (Instance variables)
5) Method Headers
6) Void Methods
7) Return Methods

Unit 3: What is a runner class?

1) Creating a runner class
2) Creating objects from a class (creating 'instances' of a class)
3) The reference variable vs. the object
4) The difference between primitive variables and object variables
5) Calling methods on an object
6) Changing the 'state' of an object
7) Multiple objects from the same class

Concept 2: Advanced topics involving Java Classes

Unit 1: Learning more about Constructors

1) Overloading a constructor
2) The keyword 'this'

Unit 2: Learning more about Methods

1) Methods that take parameters
2) Overloading a method
3) Method signatures (multiple parameter types)
4) When can a method really change the value of a variable? (primitive vs. object data)
5) Understanding the toString() method

Unit 3: Learning more about Objects

1) The null reference
2) The word 'new' calls the constructor
3) Review of the reference variable vs. the object

Unit 4: Learning more about Variables

1) Static variables
2) Final variables
3) Scope of local variables and instance variables

Unit 5: Learning more about Classes and variables

1) Static classes

Concept 3: Common Classes that you must be familiar with

Unit 1: The Object class (The mother of all Classes)

1) Every object knows how to…

Unit 2: The String class

1) Ways to make a String object
2) Concatenating Strings
3) Comparing String Objects
4) Important String Methods

Unit 3: The Math class

1) Did you know that the Math class is a static class? What does that mean?
2) How to make Random numbers using the Math class
3) Important Math class methods

Unit 4: Wrapper classes

1) Why have Wrapper classes?
2) The Integer Class
3) The Double Class
4) The Boolean Class

Concept 4: Data, Data, Data

Unit 1: Which data structure should you use?

1) Arrays
2) ArrayLists
3) 2-Dimensional Arrays
4) The List Interface

Unit 2: Arrays

1) Declaring an Array
2) Always start at Zero
3) Length of an Array
4) An Array doesn't have methods (but Arrays does, huh?)
5) Traversing an Array
6) The for-each loop (the enhanced for-loop)
7) Arrays as Parameters
8) Array variables in a class
9) Array of Objects

Unit 3: Array Lists

1) Declaring an ArrayList
2) Always start at Zero
3) Length of an ArrayList
4) Important ArrayList methods
5) ArrayLists cannot hold primitive data types
6) Traversing an Array
7) The for-each loop (the enhanced for-loop)
8) ArrayLists as Parameters
9) ArrayList variables in a Class

10) An ArrayList of Class Objects
11) The List<E> interface
12) Storing data to a .txt file (FileUtility Class)

Unit 4: Two-Dimensional Arrays

1) Always (row, column)
2) Always start at Zero
3) How to get the number of rows
4) How to get the number of columns
5) Traversing a 2-dimensional array
6) Row-major order
7) Column-major order

Concept 5: Searching and Sorting

Unit 1: Searching for an item

1) Searching for an item in an array (sequential search)
2) Searching for an item in an ArrayList (sequential search)
3) Brute Force

Unit 2: Sorting Data

1) What is efficiency?
2) Sorting an array
3) Sorting an ArrayList
4) Swapping data in an array
5) Swapping data in an ArrayList
6) The Bubble Sort
7) The Selection Sort
8) The Insertion Sort
9) The Merge Sort

Unit 3: The Binary Search

1) The data needs to be sorted first
2) The best way to win the Guessing Game

Concept 6: Advanced Java Design Structures

Unit 1: Extending a class

1) Inheritance among a superclass and a subclass
2) Parent-Child relationship
3) The inheritance hierarchy
4) How to use the keyword 'super'
5) Who 'Is-a' what?
6) Who 'Has-a' what?
7) Data encapsulation – private vs. public data
8) Who's calling whom? Overriding methods
9) Who knows what? Instance variables in the hierarchy

Unit 2: The Interface

1) A Contract with the classes that implement it
2) Anatomy of an interface
3) Unimplemented methods
4) Example: The Comparable interface

Unit 3: Polymorphism

1) Why do I act the way I do?
2) Dynamic Binding (Late Binding)

Unit 4: Abstract classes

1) Anatomy of an Abstract Class
2) Implemented methods
3) Unimplemented methods
4) Just a minute, I'm not quite done

Unit 5: Type Compatibility

1) Downcasting
2) The ClassCastException

Concept 7: Program Design and Analysis

Unit 1: The Software Development Cycle

1) The Waterfall Model
2) Program Specification
3) Program Design
4) Program Implementation
5) Testing and Debugging
6) Program Maintenance

Unit 2: Object-Oriented Design

1) Identifying Classes
2) Identifying Behaviors
3) Determining Relationships between Classes
4) UML Diagrams

Unit 3: Program Analysis

1) Program Correctness
2) Assertions
3) Pre-Conditions
4) Post-Conditions
5) Efficiency

Concept 8: Recursion

Unit 1: What is a recursive method?

1) The general form for of simple recursive methods
2) The Base Case
3) Writing your own recursive methods

Concept 9: The GridWorld Case Study

Unit 1: Playing around with Gridworld

1) Running the BugRunner Class
2) Modifying code in BugRunner
3) Checking out the existing methods

Unit 2: The main players in Gridworld

1) The Actor Class
2) The Bug Class
3) The Rock Class
4) The Flower Class
5) The Critter Class

Unit 3: Support Classes and Interface

1) The Location Class
2) The Grid interface
3) The Abstract Grid Class

4) The Bounded Grid
5) The Unbounded Grid
6) The 'Black Box' classes
7) UML Diagram of Gridworld

Unit 4: Extending Classes
1) Extending the Actor Class
2) Extending the Bug Class
3) Extending the Critter Class

Concept 10: Building a Custom Application with a Small Group

Unit 1: Find a need
1) Explore options
2) Find a client

Unit 2: Design, Design, Design
1) Initial design of 'look' of the application
2) Creating a Mock-Up
3) Making an application user-friendly
4) Initial design of the classes
5) Initial design of the methods for each class
6) Initial design of the data structures
7) Initial design of hierarchy (Superclasses & Subclasses)
8) Initial design of interfaces & abstract classes
9) Create a list of clarifications needed from the client

Unit 3: Meet with the client
1) Present the initial design to the client
2) Ask for further requirements from the client
3) Clear up any questions between you and the client

Unit 4: Revise Initial Design
1) Make appropriate design changes
2) Further clarify the purpose of each class and method

Unit 5: Implementation
1) Divide the work load based on strengths and weaknesses
2) Clearly communicate the goals for each person
3) Set deadlines for each person

4) Inform members of the group of major changes to design
5) Reassign work as needed among group members

Unit 6: Testing

1) Debug by considering all basic data first
2) Consider obscure data and decide how to handle it
3) Test storage and retrieval of data in the .txt file

Unit 7: Presentation to the Client for Review

1) Present the application to the client
2) Request honest feedback and take notes
3) Give client realistic feedback on specific requests (we can do this, but we can't do that)

Unit 8: Final Revisions

1) Make appropriate modifications to the application that respond to client's feedback
2) Consider final improvements to make it 'pretty'

Unit 9: Final Presentation to the Client

1) Present the application to the client
2) Provide in-service to client on how to run the application
3) Provide commitment by providing email and cell

Unit 10: Application Maintenance

1) Be available to support your client
2) Provide application maintenance by modifying incorrect code

Requirements for Large Client-Based Application	**Completed**
Phase 1: Find a need	
Phase 2: Design, Design, Design	
Phase 3: Meet with the client	
Phase 4: Revise Initial Design	
Phase 5: Implementation	
Phase 6: Testing	
Phase 7: Presentation to the Client for Review	
Phase 8: Final Revisions	
Phase 9: Final Presentation to the Client	
Phase 10: Application Maintenance	
Project Completed	